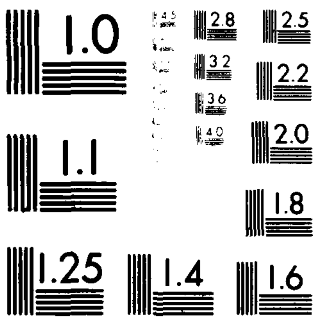
1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

LEVEL II

ARO 164.70.15M

13

Systems
Optimization
Laboratory

Department of Operations Research
Stanford University
Stanford, CA 94305

(1)

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

(6)

QP-BASED METHODS FOR LARGE-SCALE NONLINEARLY CONSTRAINED OPTIMIZATION.

by

(10) **Philip E. Gill, Walter Murray,**
Michael A. Saunders and Margaret H. Wright

(12) [illegible]

(9) **TECHNICAL REPORT, SOL 81-1**

[illegible]

(11) **January 1981**

N00014-75-[illegible]
DAAG[illegible]9-79-[illegible]

[illegible]

Research and reproduction of this report were supported by the Department of Energy Contract DE-AC03-76SF00326, PA No. DE-AT03-76ER72018, Office of Naval Research Contract N00014-75-C-0267, National Science Foundation Grants MCS-7926009 and ECS-8012974, and Army Research Office Contract DAAG29-79-C-0110.

JOB

[illegible] **81 4 6 229**

QP-BASED METHODS FOR LARGE-SCALE NONLINEARLY CONSTRAINED OPTIMIZATION†

by

Philip E. Gill, Walter Murray,
Michael A. Saunders and Margaret H. Wright

Systems Optimisation Laboratory
Department of Operations Research
Stanford University
Stanford, CA 94305

January 1981

ABSTRACT

Several methods for nonlinearly constrained optimisation have been suggested in recent years that are based on solving a quadratic programming (QP) subproblem to determine the direction of search. Even for dense problems, there is no consensus at present concerning the "best" formulation of the QP subproblem. When solving large problems, many of the options possible for small problems become unreasonably expensive in terms of storage and/or arithmetic operations. This paper discusses the inherent difficulties of developing QP-based methods for large-scale nonlinearly constrained optimisation, and suggests some possible approaches.

This research was supported by the U.S. Department of Energy Contract DE-AC03-76SF00326, PA No. DE-AT03-76ER72018; National Science Foundation Grants MCS-7926009 and ECS-[illegible]12974; the Office of Naval Research Contract N00014-75-C-0267; and the U.S. Army Research Office Contract DAAG29-79-C-0110.

† Presented at the Nonlinear Programming 4 Symposium, Madison, Wisconsin, July 1980.

1. Introduction

The problem of concern is the following:

$$\boxed{\begin{array}{lll} \text{NCP} & \underset{x\in\mathbb{R}^n}{\text{minimise}} & F(x) \\ & \text{subject to} & Ax = b \\ & & c(x)\left\{\begin{smallmatrix} = \\ \geq \end{smallmatrix}\right\}0 \\ & & \ell \leq x \leq u. \end{array}}$$

The objective function $F(x)$ is assumed to be twice-continuously differentiable. The matrix A has m_1 rows; the vector $c(x)$ contains a set of twice-continuously differentiable nonlinear constraint functions $\{c_i(x)\}$, $i = 1, \ldots, m_2$.

We assume that the number of variables and constraints in NCP is "large", and that A is sparse. Obviously, the definition of "large" depends on the available storage and computation time. It will generally be assumed that the number of nonlinear constraints is small relative to the number of linear constraints.

No general linear *inequality* constraints have been included in the form NCP because the methods to be discussed are based on extensions of the simplex method (see, e.g., Dantzig, 1963). In solving large linear programs (LPs), inequality constraints are converted to equalities by adding slack variables. The purpose of this transformation is to allow the simplex method to be implemented with only column operations on the constraint matrix. Furthermore, since A is stored in compact form, the added slack variables do not significantly increase the storage requirements.

There is still no universal consensus among researchers about the "best" algorithm for nonlinearly constrained optimisation in the dense case. However, it is generally agreed that methods based on a quadratic programming (QP) subproblem are very effective (one class of such methods will be briefly summarised in Section 3). Our concern in this paper is with the general effect of problem size on the algorithmic procedures associated with QP-based methods, rather than with a complete description of a particular algorithm. We shall consider the mechanics of the computations and the modifications that are necessary to perform them efficiently (or at all).

As a rule, there are *fewer* algorithmic options for large problems, since many computational procedures that are standard for small problems become unreasonably expensive in terms of arithmetic and/or storage. However, in another sense the options for large problems are less straightforward because of the critical effect on efficiency of special problem structure and the details of implementation.

When solving large problems, it may be necessary to alter or compromise what seems to be an "ideal" or "natural" strategy. In fact, an approach that would not be considered for small problems may turn out to be the best choice for some large problems. For example, in solving large LP problems by the simplex method, it is often very expensive to compute *all* the Lagrange multipliers in order to choose the incoming column at a given iteration. With a "partial pricing" strategy (see, e.g., Orchard-Hays, 1968), only some of the multipliers are computed. Although more iterations may be required to obtain the optimal solution, the work per iteration is typically lower, and thus the total computational effort may be decreased.

Similarly, certain standard assumptions about the relative costs of portions of an algorithm become invalid in the large-scale case. For example, the measure of efficiency of an algorithm for dense unconstrained optimization is often taken as the number of evaluations of user-supplied functions (e.g., the objective function, the gradient) that are required to reach a specified level of accuracy. Although this measure is recognised to be overly simplistic (see, e.g., Hillstrom, 1977; Lyness and Greenwell, 1977), it is nonetheless a reasonable measure of effectiveness for most problems. This is because the number of arithmetic operations per iteration tends to be of order n^3 at most, and the amount of work required for storage manipulation is negligible. However, even for unconstrained problems of moderate size, the work associated with linear algebraic procedures and data structure operations tends to become significant with respect to the function evaluations (see, e.g., the timing results obtained by Thapa, 1980).

The following assumptions and notation will be used throughout the paper. A local minimum of NCP will be denoted by x^*. The gradient of $F(x)$ will be denoted by $g(x)$, and its Hessian matrix by $G(x)$. The gradient of the i-th nonlinear constraint function $c_i(x)$ will be denoted by $a_i(x)$, and its Hessian by $G_i(x)$. The first-order Kuhn-Tucker conditions (see, e.g., Fiacco and McCormick, 1968) will be assumed to hold at x^*, so that there exists a Lagrange multiplier vector λ^* corresponding to the active constraints.

In an iterative method for computing x^*, the $(k+1)$-th iterate is defined as

$$x_{k+1} = x_k + \alpha_k p_k,$$

where p_k is the *search direction* and the positive scalar α_k is the *step length*. Usually, p_k is chosen to be a descent direction with respect to some merit function, and α_k is chosen to produce a "sufficient decrease" in the merit function (see Ortega and Rheinboldt, 1970, for a definition of "sufficient decrease").

2. Large-Scale Linearly Constrained Optimization

In this section, we briefly review the key features of an efficient method for large-scale linearly constrained optimization. The problem format is given by

$$\begin{aligned} \underset{x\in\Re^n}{\text{minimise}} \quad & F(x) \\ \text{subject to} \quad & Ax = b \\ & \ell \le x \le u. \end{aligned} \tag{1}$$

The algorithm for (1) to be described is the reduced-gradient algorithm (Wolfe, 1962) of Murtagh and Saunders (1978), which has been implemented in the Fortran program MINOS (Murtagh and Saunders, 1977). An "active set" strategy is used to compute the search direction; this means that at each iteration a certain subset of the bounds are treated as equalities, and the corresponding variables are held fixed at these bounds during the iteration. Let $\hat{A}$ denote the matrix of coefficients corresponding to the active constraints; $\hat{A}$ will contain all the general constraints plus the active bounds. In order for the same constraints to be active at the next iterate, the search direction must satisfy

$$\hat{A}p = 0. \tag{2}$$

Let Z denote a matrix whose columns form a basis for the null space of $\hat{A}$, so that $\hat{A}Z = 0$. The relationship (2) implies that p must be a linear combination of the columns of Z, i.e.,

$$p = Zp_z \tag{3}$$

for some p_z.

In order to define a direction that satisfies (2) when A is large and sparse, the matrix A is (conceptually) partitioned as follows:

$$A = (B \quad S \quad N). \tag{4}$$

The matrix B (for "basis", by analogy with linear programming) is square and non-singular, and its columns correspond to the *basic* variables. The columns of N correspond to the *nonbasic* variables (those to be fixed on their bounds). The columns of the matrix S correspond to the remaining variables, which are termed *superbasic*. Note that the number of columns in B is fixed, but the numbers of columns in S and N may vary. We emphasise that only *column* operations are performed on B as the algorithm proceeds.

At a given iteration, the active constraints are given by

$$\begin{pmatrix} B & S & N \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} x_B \\ x_S \\ x_N \end{pmatrix} = \begin{pmatrix} b \\ b_N \end{pmatrix}, \tag{5}$$

where the components of b_N are taken from either ℓ or u, depending on whether the lower or upper bound is active.

The matrix Z used here can be represented as

$$Z = \begin{pmatrix} -B^{-1}S \\ I \\ 0 \end{pmatrix}. \tag{6}$$

Naturally, B^{-1} and Z are not computed explicitly. Rather, a sparse LU factorization of B is maintained; periodic refactorization (often termed "reinversion") is used to condense storage and regain accuracy in the factors (see Saunders, 1976; Reid, 1976).

The form (6) of Z means that the partitioning of variables into basic, nonbasic, and superbasic sets carries over to the calculation of the search direction, p. If p is partitioned as $(p_B \; p_S \; p_N)$, from (3) and (6) we see that $p_N = 0$ and

$$Bp_B = -Sp_S. \tag{7}$$

Equation (7) shows that p_B can be computed in terms of p_S, and thus the superbasic variables act as the "driving force" in the minimization. To determine the vector p_S, a quadratic approximation to the objective function is minimized subject to the constraint (2); the search direction therefore "solves" an equality-constrained quadratic program of the form

$$\underset{p\in\Re^n}{\text{minimize}} \quad \frac{1}{2}p^THp + g^Tp \tag{8a}$$

$$\text{subject to} \quad \hat{A}p = 0, \tag{8b}$$

where H is an approximation to $G(x)$, the Hessian matrix of $F(x)$. The solution of (8) can be obtained using only the projected matrix Z^THZ (so that H itself is not required). In the Murtagh-Saunders algorithm, a quasi-Newton approximation to $Z^TG(x)Z$ is maintained in the factorized form R^TR, where R is upper triangular, and p_S is computed from

$$R^TRp_S = -Z^Tg = -S^TB^{-T}g_B + g_S. \tag{9}$$

After p_S and p_B have been computed from (9) and (7), the value of the step length is chosen to achieve a suitable reduction in F.

As long as $\|Z^Tg\|$ is "large", only the basic and superbasic variables are optimized. If one of these variables encounters a bound as the iterations proceed, it is moved into the set of nonbasic variables, and the set of active constraints is altered acordingly.

When $\|Z^Tg\|$ is "small", it is considered that the current iterate is "nearly" optimal on the current set of active constraints. In this situation, we determine whether the objective function can be further reduced by releasing any nonbasic variable from its bound. This possibility is checked by computing Lagrange multiplier estimates from the system

$$\begin{pmatrix} B^T & 0 \\ S^T & 0 \\ N^T & I \end{pmatrix} \begin{pmatrix} \pi \\ \sigma \end{pmatrix} = \begin{pmatrix} g_B \\ g_S \\ g_N \end{pmatrix}. \tag{10}$$

We define the vectors π and σ from

$$B^T\pi = g_B; \tag{11}$$

$$\sigma = g_N - N^T\pi. \tag{12}$$

The system (10) is compatible when $Z^Tg = 0$, since in this case

$$g_S = S^TB^{-T}g_B = S^T\pi.$$

The vector σ thus provides a set of Lagrange multipliers for the bound constraints that are active on nonbasic variables. If a nonbasic variable can be released from its bound, the iterations continue with an expanded superbasic set.

The procedures of this method differ in several ways from those used in the dense case. Firstly, the null space of $\hat{A}$ is defined in terms of a *partition of the variables*, rather than a matrix Z with orthogonal columns (see Gill and Murray, 1974). The expression (6) for Z indicates that an ill-conditioned basis matrix B can affect the condition of all calculations in the algorithm, and may drastically alter the scaling of the variables. When the columns of Z are orthogonal, $\|Z^Tg\|_2 \leq \|g\|_2$; otherwise, Z^Tg is "unscaled". Since an orthogonal Z is not practical (in terms of storage or computation) for most large-scale problems, additional numerical difficulties are likely in the computation of p and the projected Hessian approximation.

Secondly, the multiplier estimates computed from (10) are exact only when $Z^Tg = 0$, and the neighborhood in which their sign is correct depends on the *condition of B*. Hence, when $\|Z^Tg\|$ is merely "small", it may be inefficient to release a variable based on the vector σ from (12). Although a feasible descent direction can be computed, the deleted constraint may very soon be re-encountered. This difficulty is less severe in the dense case, where the multiplier estimates computed with an orthogonal Z will in general have the correct sign in a much larger neighborhood of a constrained stationary point because the size of the neighborhood depends only on the condition of $\hat{A}$ (see Gill and Murray, 1979a, for further discussion). The increased unreliability of Lagrange multiplier estimates is unavoidable when Z is given by (6), and must be taken into account in all large-scale optimization.

Finally, the cost of computing and updating the factorisation of B is substantial, in terms of both arithmetic operations and storage manipulation. For many large-scale problems, the work associated with performing the steps of the algorithm completely dominates the cost of evaluating the nonlinear objective function.

3. QP-Based Methods for Dense Problems

This section will be concerned entirely with the treatment of *nonlinear* constraints. If the problem NCP contains only nonlinear constraints, the assumed optimality conditions imply that x^* is a stationary point of the *Lagrangian function*

$$L(x,\lambda) \equiv F(x) - \lambda^T\hat{c}(x),$$

when $\lambda = \lambda^*$ (where $\hat{c}(x)$ denotes the set of constraints that hold with equality at x^*).

In a QP-based method, the search direction is the solution of a QP subproblem

$$\underset{p\in\Re^n}{\text{minimise}} \quad \frac{1}{2}p^THp + g_L^Tp \tag{13a}$$

$$\text{subject to} \quad Ap\begin{Bmatrix} = \\ \geq \end{Bmatrix}d. \tag{13b}$$

The quadratic objective function (13*a*) of the QP subproblem is often viewed as a quadratic approximation to the *Lagrangian function*, in which case g_L and H represent the gradient and Hessian of the Lagrangian function, respectively. However, the vector g_L is usually taken as $g(x_k)$; this choice does not alter the solution of some QP subproblems, and has the benefit that the multipliers of the subproblem may then be used as estimates of the multipliers of NCP. The linear constraints (13*b*) of the subproblem are based on linearisations of the nonlinear constraints at the current point, and thus A in (13*b*) usually includes the constraint gradients $\{a_i(x_k)\}$.

QP-based methods have been proposed by many, including Wilson (1963), Murray (1969), Biggs (1972), Garcia and Mangasarian (1976), Han (1976, 1977), Wright (1976), and Powell (1977, 1978). Although it can be shown that under certain conditions, QP-based methods are equivalent to other methods (Tapia, 1978), we shall consider only methods in which a QP subproblem is actually solved to obtain the search direction.

There are substantial variations in formulation of the QP subproblem (13). Certain crucial issues remain unresolved: representation of H in (13*a*); specification of d in (13*b*); treatment of an infeasible or unbounded subproblem;

recovery from ill-conditioning; computation of reliable Lagrange multiplier estimates; definition of a merit function to be used in defining the step length α_k; and maintenance (if possible) of superlinear convergence. See Maratos (1978), Chamberlain (1979), Murray and Wright (1980) and Chamberlain *et al.* (1980) for discussions of some of these issues.

We shall mainly discuss a strategy based on formulating the QP subproblem (13) with only *equality* constraints. This will be termed the equality QP (EQP) approach, and the subproblem is given by

$$\underset{p\in\Re^n}{\text{minimise}} \quad \frac{1}{2}p^THp + g^Tp \tag{14a}$$

$$\text{subject to} \quad \hat{A}p = \hat{d}. \tag{14b}$$

The t rows of the matrix $\hat{A}$ represent a selection of constraints that are considered to be "active". (We shall not be concerned with *how* these constraints are selected.) The constraints (14*b*) are assumed to be compatible.

It is convenient both conceptually and computationally to write the solution of (14) as the sum of two orthogonal vectors. Let Y denote a matrix whose columns form a basis for the *range space* of $\hat{A}^T$; as in Section 2, Z will denote a matrix whose columns form a basis for the *null space* of $\hat{A}$. The solution of (14) can be written as

$$p^* = Yp_Y + Zp_Z. \tag{15}$$

The vector p_Y is the solution of the linear system

$$\hat{A}Yp_Y = \hat{d}. \tag{16}$$

When the constraints (14*b*) are consistent, the system (16) is compatible; when $\hat{A}$ has full row rank, the vector p_Y is unique, since $\hat{A}Y$ is a non-singular matrix. An important difference from the linear-constraint case is that in general p_Y is *non-zero* (since $\hat{d}$ is non-zero). Hence, we must take account of the range space component as well as the null space component in computing the search direction.

The vector p_Z is the solution of the linear system

$$Z^THZp_Z = -Z^Tg - Z^THYp_Y. \tag{17}$$

When Z^THZ is indefinite, the interpretation of (17) is ambiguous; see Murray and Wright (1980) for further discussion of this point.

When $\hat{A}$ is small, suitable matrices Y and Z with orthonormal columns can be obtained from the LQ factorization of $\hat{A}$, since we have

$$\hat{A}Q = \hat{A}(\,Y \quad Z\,) = (\,L \quad 0\,).$$

The $(n-t) \times (n-t)$ matrix $Z^T H Z$ needed to compute p_z can be formed in various ways. If second derivatives are available, H can be taken as W, the current approximation of the Hessian of the Lagrangian function

$$W = G(x_k) - \sum_{i=1}^{t} \lambda_i G_i(x_k),$$

where $\{\lambda_i\}$ are the current Lagrange multiplier estimates. If first derivatives are available, the matrix WZ can be approximated by finite-differences of the gradient of the Lagrangian function along the columns of Z. Note that the matrix HZ (rather than H itself) is required to solve (17); thus, substantial efficiencies are possible with a discrete Newton method, since only $n-t$ gradient evaluations are required to approximate WZ, compared to the possible n evaluations needed to approximate the full matrix W. A quasi-Newton approximation to W or $Z^T W Z$ may also be recurred.

4. The Use of a Linearly Constrained Subproblem

Given the sophisticated techniques available for large-scale linearly constrained optimization (see Section 2), it is logical to attempt to apply them to the nonlinearly constrained problem NCP. One possible way to do so is to pose a sequence of linearly constrained subproblems with a general (rather than quadratic) objective function. Such a method was proposed by Robinson (1972) and Rosen and Kreuser (1972), and more recently by several others (e.g., Van der Hoek, 1979). The specific application of this idea to large-scale problems using the algorithm described in Section 2 has been suggested by Rosen (1978) and Murtagh and Saunders (1980a,b).

In order to give the flavor of this approach, we shall briefly describe the algorithm of Murtagh and Saunders. Let x_k and λ_k denote the current iterate and the current estimate of the Lagrange multipliers; other quantities subscripted by k will denote those quantities evaluated at x_k. The next iterate is obtained by solving the linearly constrained subproblem

$$\begin{array}{ll} \underset{x \in \Re^n}{\text{minimize}} & F(x) - \lambda_k{}^T \bar{c}(x) + \frac{1}{2}\rho \bar{c}(x)^T \bar{c}(x) \\ \text{subject to} & Ax = b \\ & \bar{A}_k(x - x_k) \left\{ \begin{matrix} = \\ \geq \end{matrix} \right\} - c_k \\ & \ell \leq x \leq u. \end{array} \tag{18}$$

The constraints of (18) that are involved in $\bar{A}_k$ are obtained by linearizing all the nonlinear constraints. The function $\bar{c}(x)$ is defined as the original nonlinear function minus its current linearisation:

$$\bar{c}(x) = c(x) - c_k - \tilde{A}_k(x - x_k).$$

Note that any iterative procedure for solving (18) requires evaluation of the problem functions (in contrast to solving a QP subproblem, where all the work is linear-algebraic).

The nonlinear objective function of the subproblem (18) is called a *modified augmented Lagrangian function*. The penalty term $\frac{1}{2}\rho\bar{c}(x)^T\bar{c}(x)$ is included to encourage progress from a poor starting point. When x_k is judged to be sufficiently close to x^*, the penalty parameter ρ is set to zero in order to achieve the quadratic convergence proved by Robinson (1972, 1974).

Some aspects of this approach to solving NCP are relevant to our later discussion of QP-based methods. Other aspects illustrate the compromises that are often necessary in solving large-scale problems.

For dense problems, methods based on linearly constrained subproblems have generally been regarded as less efficient than QP-based methods, in terms of the number of function evaluations required for convergence (see, e.g., the comments in Murtagh and Saunders, 1980a). For certain problem categories, solving a more difficult subproblem sometimes leads to an improvement in overall efficiency. However, the additional work necessary to solve (18) as compared to a QP does not appear to produce a comparable increase in efficiency for problems in which the overhead associated with performing linear algebraic procedures is small relative to the cost of evaluating the nonlinear functions. Whether the tradeoff will be different in the sparse case is still unknown.

A method based on (18) generates the next "outer" iterate through a subproblem whose solution also requires an iterative procedure (which generates "inner" iterates). In the MINOS/AUGMENTED implementation of this method (Murtagh and Saunders, 1980b), a limit is imposed on the number of inner iterations. If the maximum number of such iterations is reached, the inner procedure is terminated, and its final iterate is taken as the next outer iterate. It seems essential to impose such a limit in the large-scale case, since it is unlikely that the initial Jacobian approximation and multiplier estimates will remain appropriate if hundreds of iterations are required to reach optimality for the subproblem. The effects of such premature termination remain to be analysed.

A "good" choice for the penalty parameter ρ is crucial in the success and efficiency of the method on certain problems. The considerations in selecting ρ are similar to those in an augmented Lagrangian method (for a survey, see Fletcher 1974). A too-small value of ρ may lead to excessive constraint violations in the solution of (18), an unbounded subproblem (18), or a poorly conditioned Hessian of the augmented function. A too-large value of ρ may also cause the Hessian to be ill conditioned; it may have the additional undesirable effect of forcing the iterates to follow the constraint boundary very closely. Furthermore, the decision as to when to set ρ to zero is not straightforward.

The value of λ_k in (18) is taken as the multiplier vector of the previous subproblem. If the previous subproblem was solved to optimality, this ensures that the multipliers corresponding to inequality constraints have the correct sign, and that $\lambda_i = 0$ for inactive constraints regardless of the partition $(B\ \ S)$. However, it means that multiplier estimates are not computed with the most recent information, but rather are based on the "old" Jacobian. In addition, the interpretation of the available Lagrange multiplier estimates is further complicated if an inner iteration is terminated before convergence.

5. Extension of QP-Based Methods to the Large-Scale Case

In the remainder of this paper, we shall consider some of the issues in developing a QP-based method such as those described in Section 3 for the problem NCP. It is assumed that the sparsity pattern of the Jacobian of the nonlinear constraints is known a *priori*.

Even for dense problems, linear constraints (especially bounds) should be treated separately from nonlinear constraints. If they are not, considerable inefficiencies tend to be introduced into solution methods; furthermore, the iterates will not in general be feasible with respect to the linear constraints. We believe that QP-based methods should treat any linear constraints as if they were the only constraints in the problem, in order to take advantage of the efficiencies associated with purely linear constraints, and to ensure that the nonlinear functions are evaluated only at points that satisfy the linear constraints.

When this approach is taken in an EQP method, the matrix $\hat{A}$ in (14*b*) will include the general linear constraints and active bounds as well as the current gradients of the nonlinear constraints. An EQP method should therefore include a strategy to exploit the fact that only *part* of $\hat{A}$ changes from one iteration to the next, since the rows of $\hat{A}$ that correspond to linear constraints and simple bounds remain constant.

In the dense version of the EQP method discussed in Section 3, the search direction was represented (and computed) in terms of matrices Y and Z obtained from the LQ factorisation of $\hat{A}$. A similar representation of the solution of (14) must be developed that is suitable for large-scale problems. In this section, we sketch one possible approach, which is similar to that described in Section 2 for the large-scale linear-constraint case.

The variables are partitioned into basic, superbasic, and nonbasic sets, with a corresponding partition of the columns of $\hat{A}$ and the components of p and g. Since the nonbasic variables are fixed on their bounds during a given iteration, the vector p_N must be zero (and can be ignored). To satisfy the linear constraints

(14*b*), it must hold that

$$\begin{pmatrix} B & S \end{pmatrix}\begin{pmatrix} p_B \\ p_S \end{pmatrix} = d, \qquad (19)$$

where B is a $t \times t$ square non-singular matrix, and d contains appropriate elements of $\hat{d}$. From (19) it follows that

$$Bp_B = d - Sp_S. \qquad (20)$$

Note that the components of d will be zero in positions corresponding to linear constraints. Hence, for *any* p_S, the definition of p_B by (20) ensures that p will be feasible with respect to the linear constraints of both the subproblem and the original problem.

The vector p_S is determined by minimization of the quadratic objective function (14*a*). Writing this objective in terms of the partitioned vector p, we obtain

$$\begin{aligned}\frac{1}{2}p_B^T H_B p_B + p_B^T H_{BS} p_S + \frac{1}{2}p_S^T H_S p_S \\ + g_B^T p_B + g_S^T p_S,\end{aligned} \qquad (21)$$

where

$$H = \begin{pmatrix} H_B & H_{BS} \\ H_{BS}^T & H_S \end{pmatrix}.$$

Substituting for p_B using (20) makes (21) a quadratic function in p_S alone. The optimal p_S is the solution of a system of equations exactly analogous to (17) for the dense case:

$$Z^T H Z p_S = -Z^T g + Z^T H \begin{pmatrix} B^{-1}d \\ 0 \end{pmatrix}, \qquad (22)$$

where Z is given by (6).

At a typical iteration, B is given by

$$B = \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix} \begin{matrix} \}t_1 \\ \}t_2 \end{matrix}. \qquad (23)$$

If we assume that the linear constraints are placed first, the first t_1 rows (the matrices B_1 and B_2) correspond to the linear constraints, and the last t_2 rows (the matrices B_3 and B_4) correspond to the nonlinear constraints. Both B_1 and B_4 are square.

With the EQP approach described in Section 3, the matrix $\hat{A}$ in (14) includes only the gradients of the *active* nonlinear constraints. The question therefore arises of how to treat nonlinear inequalities in large-scale problems. The reasons noted in Section 1 for adding slack variables to linear inequality constraints also apply to nonlinear inequality constraints. However, there might appear to be some disadvantages. In particular, keeping all nonlinear constraints as rows of B would seem to result in increased housekeeping costs, as well as wasted computational effort in computing the gradients of inactive constraints. In fact, neither of these disadvantages applies, and the slack variables corresponding to nonlinear inequalities can be included with very little cost. The elements of the search direction corresponding to the slack variables of the nonlinear constraints can be ignored, and the line search involves only the original variables. The value of a nonlinear slack variable at the next iterate is given by the *recomputed* constraint value, which is used to determine whether the slack variable is basic. All the other coefficients in the row of the Jacobian associated with a basic slack variable can be set to zero, and there is no need to compute the gradient of the corresponding constraint.

6. Representing the Basis Inverse

In this section, we consider methods for representing B^{-1} as the iterations of an EQP method proceed. The inverse is *never* represented explicitly. However, we use this terminology because the methods to be described solve the linear systems that involve B without a complete factorisation of B.

Changes in the columns of B that result as variables move on and off bounds can be carried out exactly as in the linear-constraint case. The difficulty in a nonlinearly constrained problem is that the last t_2 *rows* of B will change at each iteration due to constraint nonlinearities. We assume that it is not computationally feasible to refactorise B at every iteration; however, periodic refactorisation will be performed to condense storage and ensure accuracy in the factors.

If both t_2 and the number of non-zero elements in the last t_2 rows of B are small, the changes in B due to constraint nonlinearities represent only a small number of column changes. In this case, it would be practical to update the LU factors of B in a standard fashion (see, e.g., Forrest and Tomlin, 1972; Reid, 1976; Saunders, 1976). However, each iteration would involve several column updates, and hence refactorisation would be required at more frequent intervals.

6.1. Partitioning. Since B_1 includes only linear constraints, it is possible to recur a factorisation of B_1 from iteration to iteration. This fact can be utilised to advantage because systems of equations involving B or B^T can be solved using factorisations of B_1 and a matrix the size of B_4.

For example, if the vector b is partitioned corresponding to (23) as $(b_1 \;\; b_2)$, the solution of $Bx = b$ can be represented as

$$x = \begin{pmatrix} u_1 + u_2 \\ v_1 \end{pmatrix},$$

where the vectors u_1, u_2 and v_1 are calculated from

$$\begin{aligned} B_1 u_1 &= b_1, \\ D v_1 &= b_2 - B_3 u_1, \\ B_1 u_2 &= -B_2 v_1, \end{aligned} \tag{24}$$

where

$$D = B_4 - B_3 B_1^{-1} B_2. \tag{25}$$

This procedure is sometimes described as a *partitioned inverse technique* (see, e.g., Faddeeva, 1959). The matrix (25) is called the *Schur complement* (see, e.g., Cottle, 1974). The steps of (24) are equivalent to block Gaussian elimination on B, with B_1 as the first block.

If $t_1 \gg t_2$, the main work in (24) involves obtaining $B_1^{-1}B_2$ (or $B_1^{-T}B_3^T$) in (25). To reduce the work in this calculation, it is helpful to maximise the number of zero columns of B_2 and/or B_3^T. This can be borne in mind whenever B_1 is refactorised, since there are some degrees of freedom in deciding which variables are to be basic and superbasic. Once the LU factors of B_1 are available, the matrices needed to compute D can be obtained by forming $L^{-1}B_2$ and $U^{-T}B_3^T$. In the large-scale case, however, it will usually be more efficient to compute $U^{-1}L^{-1}B_2$ or $L^{-T}U^{-T}B_3^T$, depending on whether L and U are stored by columns or rows.

Although B_1 is required to be a fixed size with this approach, the number of active *nonlinear* constraints may vary. Therefore, it is not necessary to include slack variables for the nonlinear inequality constraints.

6.2. An Approximate Inverse; Iterative Improvement. An obvious strategy for overcoming the difficulties of updating B^{-1} as its last rows change is simply not to update it. The technique of retaining a constant Jacobian or Hessian approximation in Newton-type methods is widely used (see, e.g., Dennis, 1971), and has been thoroughly analysed. With the approach described in Section 4, the linear constraints remain constant until the general subproblem (18) has been solved.

This idea and its extensions can be applied to a QP-based method in several ways. Let $\bar{B}^{-1}$ denote an available representation of an *approximation* to the inverse of B (e.g., from the most recent factorization or some previous iteration). We shall mention two possible strategies for using $\bar{B}^{-1}$ to "solve" systems of equations such as $Bx = b$. Firstly, we can simply solve the system using $\bar{B}^{-1}$; in effect, this involves substituting $\bar{B}$ for B during some number of consecutive iterations. Secondly, $\bar{B}^{-1}$ could be used further in an *iterative improvement* procedure (see, e.g., Wilkinson, 1965), assuming that B is also available.

Such approximations are acceptable in QP-based methods because the linear constraints of the QP subproblem are typically derived from optimality conditions, and the precise specification of the linear constraints is critical only near the solution. Consequently, there is substantial freedom to define the constraints (14b) when x_k is not close to x^*, provided that a sufficient decrease in the merit function can be guaranteed.

When $\bar{B}$ is the basis matrix from a previous iteration, the error in the approximate inverse is of a special form because the first t_1 rows of B are constant. In general, $\bar{B}$ satisfies

$$\bar{B} = B + F = B + \begin{pmatrix} 0 \\ \Delta \end{pmatrix},$$

where the matrix Δ represents the change in gradients of the nonlinear constraints. Therefore, we have

$$B\bar{B}^{-1} = I + \begin{pmatrix} 0 \\ E \end{pmatrix} \begin{matrix} \}t_1 \\ \}t_2 \end{matrix}.$$

Because of the relationship between B and $\bar{B}$, the structure of the error in the approximate inverse is such that the equations (14b) corresponding to *linear* constraints are always satisfied "exactly", even if $\bar{B}$ is used rather than B. In general, p_B should satisfy

$$Bp_B = d - Sp_S.$$

If $\bar{p}_B$ is defined instead from

$$\bar{B}\bar{p}_B = d - Sp_S,$$

then it follows that

$$\begin{aligned}(B \quad S)\begin{pmatrix} \bar{p}_B \\ p_S \end{pmatrix} &= d + \begin{pmatrix} 0 \\ E \end{pmatrix}(d - Sp_S) \\ &= \begin{pmatrix} d_1 \\ d_2 + Ed - ESp_S \end{pmatrix},\end{aligned}$$

where d_1 and d_2 denote the first t_1 and last t_2 components of d, respectively. Thus, when $\bar{p}_B$ is used instead of p_B, the equalities of the QP subproblem corresponding to the *linear* constraints remain satisfied (with exact arithmetic), regardless of $\|\Delta\|$.

It is also instructive to consider the size of the error that arises from using $\bar{B}$ instead of B. Let x be the exact solution of $Bx = b$, and let h be the vector such that $\bar{B}(x + h) = b$. Assuming that $\|B^{-1}F\| < 1$, it can be shown that

$$\frac{\|h\|}{\|x\|} \leq \frac{\kappa\|F\|}{\|B\|(1 - \|B^{-1}F\|)},$$

where κ is the condition number of B. Thus, when $\|\Delta\|$ is small and B is not too ill-conditioned, the relative error in x is bounded. (Note also that the bound is independent of $\|b\|$. This is important because the right-hand side of (14b) approaches zero as the iterates converge, and it would be unacceptable for the bound on the relative error in the computed solution to increase.)

If we have the exact triangular factors L and U of $\bar{B}$ and can apply B to form the residual vector, then (with exact arithmetic) an iterative improvement procedure for solving $Bx = b$ will converge if

$$\|I - \bar{B}^{-1}B\| < 1.$$

With this approach, B must remain a fixed size, so that slack variables must be included for the nonlinear inequality constraints (in contrast to the method of Section 6.1, where only the inequalities currently considered active were included in B).

7. The Search Direction for the Superbasic Variables

Given that we can obtain a representation of B^{-1} (and hence of Z), a second issue in implementing a QP-based method for large-scale problems is how to solve the equations (22) for p_s. The difficulty is that the storage and computation associated with forming Z^THZ (or Z^TH) may be prohibitive. Since H is $n \times n$, there will in general be inadequate storage to retain a full version of H.

In many cases, the dimension of the projected Hessian matrix Z^THZ will be relatively small at every iteration, even when the problem dimension is large. If Z^THZ is small enough to be stored, standard approaches from the dense case may be used. For example, a quasi-Newton approximation of the projected Hessian of the Lagrangian function may be maintained using update procedures similar to those in the linear-constraint case. Any questions concerning such procedures apply generally to nonlinearly constrained optimization, and are not particular to large-scale problems. However, the technique of computing finite-differences along the columns of Z, which is very successful for small problems, is too expensive in the large-scale case because of the effort required to form

Z. Furthermore, even if W itself is available, it is probably too costly to form Z^TWZ.

Fortunately, another alternative is available when limitations of storage and/or computation preclude an explicit representation of Z^THZ. Although we shall discuss the method in the context of nonlinearly constrained optimization, it is equally applicable to large-scale linearly constrained optimization (e.g., in the method discussed in Section 2). Furthermore, it may be useful even when the product Z^THZ can be stored, but is too costly to compute.

The linear conjugate-gradient method (Hestenes and Stiefel, 1952) is an iterative procedure for solving the linear system

$$\bar{H}p = -\bar{g}, \tag{26}$$

where $\bar{H}$ is symmetric and positive definite, without explicitly storing the matrix $\bar{H}$. Rather, a sequence of iterates $\{p_i\}$ is generated, using only products of $\bar{H}$ with vectors. The vectors $\{p_i\}$ will be referred to as *linear iterates*, and the exact solution of (26) will be termed the *Newton direction*. The vector $\bar{g}$ is usually the gradient (or projected gradient) of some nonlinear function Φ.

Conjugate-gradient methods are relevant to solving (22) because the product of Z^THZ and a vector v can in some circumstances be computed efficiently even when Z^THZ is not available. For example, if $\bar{H}$ in (26) is of the form Z^THZ, where Z is given by (6) and H is sparse, in general $\bar{H}$ will be a dense matrix. However, if H can be retained in sparse form, and Z and Z^T can be applied as noted in Section 2, the product $\bar{H}v$ can be formed efficiently.

A sparse matrix H can be obtained in several different ways. It may happen that the Hessian of the Lagrangian function (W) is sparse, with a known sparsity pattern. (This situation is less likely than in the unconstrained case, because the Hessians of all the active constraints as well as the objective function must be sparse.) In this case, techniques are available for analyzing the sparsity pattern and determining special finite-difference vectors that permit an approximation to W to be computed with relatively few evaluations of the relevant gradients (see, e.g., Curtis, Powell and Reid, 1974; Powell and Toint, 1979). Alternatively, a sparse quasi-Newton approximation to W (see, e.g., Toint, 1977; Dennis and Schnabel, 1978; Shanno, 1979) might be developed. Although our experience with sparse quasi-Newton updates has been disappointing even in the unconstrained case (see Thapa, 1980), any improvements in such methods can be applied directly.

If the Hessian of the Lagrangian function is not sparse, it is possible to estimate the vector WZv by a finite-difference along the vector Zv. Obviously, this computation requires additional evaluations of the problem functions.

A conjugate-gradient method will be useful in solving (22) only if the linear iterates converge rapidly; by assumption, it is reasonable to compute a *relatively* *small* number of matrix-vector products involving Z^THZ. Hence, it is essential to *precondition* the conjugate-gradient method (see, e.g., Axelsson, 1977). Let C be a positive-definite symmetric matrix. The solution of (26) can be found by solving the system

$$C^{-\frac{1}{2}}\bar{H}C^{-\frac{1}{2}}y = -C^{-\frac{1}{2}}\bar{g}$$

and forming $p = C^{-\frac{1}{2}}y$. Let K denote the matrix $C^{-\frac{1}{2}}\bar{H}C^{-\frac{1}{2}}$; K has the same eigenvalues as $C^{-1}\bar{H}$, since they are similar matrices ($C^{-\frac{1}{2}}KC^{\frac{1}{2}} = C^{-1}\bar{H}$). Since the linear conjugate-gradient method is known to converge very rapidly when the coefficient matrix has clustered eigenvalues, the preconditioning matrix should be chosen so that as many as possible of the eigenvalues of $C^{-1}\bar{H}$ are close to unity.

When the *projected* Hessian is small enough to be stored explicitly, preconditioning allows second-order information to be used in conjunction with a quasi-Newton method. Thus, if a quasi-Newton approximation of Z^TWZ is maintained (e.g., as R^TR), instead of computing the quasi-Newton search direction from $R^TRp_s = -\bar{g}$ as in (9), we could solve

$$R^{-T}Z^TWZR^{-1}y = -R^{-T}\bar{g}$$

by the conjugate-gradient method, and then take the Newton direction as $p = ZR^{-1}y$. The preconditioning matrix may be modified during, or after the completion of, the iterations of the conjugate-gradient method.

The *truncated Newton method* of Dembo and Steihaug (1980) "solves" (26) by performing a *limited number* of iterations of the linear conjugate-gradient method. The final iterate of the truncated sequence is then taken as an approximate solution of (26). If a single linear iteration is used, p will be the steepest-descent direction $-\bar{g}$. Thus, the truncated Newton algorithm computes a vector that interpolates between the steepest-descent direction and the Newton direction.

Dembo and Steihaug show that, if $\bar{H}$ is positive definite and the initial iterate of the linear conjugate-gradient scheme is the steepest-descent direction $-\bar{g}$, all succeeding linear iterates will be directions of descent with respect to Φ. Gill, Murray and Nash (1981) show how to generate a sequence of descent directions for the case when $\bar{H}$ may be indefinite.

The hope with a truncated Newton method is to reduce the required number of linear conjugate-gradient steps, and the use of preconditioning would therefore seem to be essential. An additional benefit can also be produced by a preconditioning strategy. In many optimisation methods, the search direction p is computed implicitly or explicitly as

$$p = -M\bar{g},$$

where M is a positive-definite matrix; for example, limited-memory quasi-Newton methods define M as a low-rank modification to the identity matrix (see Shanno, 1978). If the matrix M is used to precondition $\bar{H}$, the vector $-M\bar{g}$ is the first member of the linear conjugate-gradient sequence, and is more likely to give a good reduction in the function than the negative gradient; see Gill, Murray and Nash (1981) for further details.

8. An Inequality QP Approach

In this section, we briefly mention an alternative formulation of the QP subproblem — as an inequality-constraint QP (IQP). (Escudero, 1980, also discusses IQP subproblems for large-scale problems.) In this case, the relational operator associated with an original nonlinear constraint is carried over to the subproblem (i.e., inequalities in the original problem become inequalities in the subproblem), and the general form of the subproblem is given by (13).

Because an IQP subproblem contains inequalities, it must be solved by an iterative QP algorithm. In general (assuming that all the variables appear nonlinearly), a full $n \times n$ matrix H must be available, since it will not be known a *priori* which set of constraints will hold as equalities at the solution. In addition, a "phase I" procedure will typically be required to find a feasible point with respect to the constraints.

All the suggestions made concerning an EQP subproblem can be applied to an IQP subproblem, since most QP algorithms are based on an active set strategy (see Cottle and Djang, 1979). Note that the two approaches differ only when more than one iteration is needed to solve the IQP. Therefore, solving an IQP subproblem is always more work than solving an EQP subproblem. As in the algorithm of MINOS/AUGMENTED, it seems essential to limit the number of iterations to be performed in solving the subproblem.

For solving a large-scale problem, an IQP approach could be implemented using a sparsity-exploiting QP method to solve (27) — for example, Tomlin's (1976) implementation of Lemke's method (Lemke, 1965). Most methods of this type are based on "pivoting" operations with the extended matrix

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix}.$$

Thus, there is a need is to develop effective variants of these methods when a *sequence of* IQP subproblems must be solved that are related in the special ways noted earlier. In particular, only the last few rows of A may vary; or H may be modified by a low-rank matrix if certain quasi-Newton techniques are used to approximate the Hessian of the Lagrangian function.

9. Conclusions

We have indicated some of the compromises necessary to implement QP-based methods for large-scale nonlinearly constrained optimisation. As in the linear-constraint case, the search direction can no longer be computed with "ideal" numerical procedures. Furthermore, it may be helpful to alter the formulation of the subproblem in the interests of computational efficiency.

It is unclear whether the superiority of QP-based methods in the dense case will carry over to large-scale problems with bounds, linear and nonlinear constraints. The alternatives now available involve a higher-level subproblem, and may be less flexible in adapting the subproblem to the unpredictability of nonlinear constraints. However, they benefit from the ability to use directly the existing codes for large-scale linearly constrained optimisation. Thus, the price paid for the greater flexibility of a QP-based method is a considerable increase in programming complexity, and a reduced ability to use existing software.

References

Axelsson, O. (1977). "Solution of linear systems of equations: iterative methods", in *Sparse Matrix Techniques* (V.A. Barber, ed.), Springer-Verlag Lecture Notes in Mathematics 572, Berlin, Heidelberg and New York.

Biggs, M. C. (1972). "Constrained minimisation using recursive equality quadratic programming", in *Numerical Methods for Non-linear Optimisation* (F. A. Lootsma, ed.), pp. 411–428, Academic Press, London and New York.

Chamberlain, R. M. (1979). Some examples of cycling in variable metric methods for constrained minimization, *Math. Prog.* 16, pp. 378–383.

Chamberlain, R. M., Lemaréchal, C., Pederson, H. C., and Powell, M. J. D. (1980). The watchdog technique for forcing convergence in algorithms for constrained optimisation, Report DAMTP 80/NA 1, University of Cambridge.

Cottle, R. W. (1974). Manifestations of the Schur complement, *Linear Alg. Applics.* 8, pp. 189–211.

Cottle, R. W. and Djang, A. (1979). Algorithmic equivalence in quadratic programming, I: a least distance programming problem, *J. Opt. Th. Applics.* 28, pp. 275–301.

Curtis, A. R., Powell, M. J. D. and Reid, J. K. (1974). On the estimation of sparse Jacobian matrices, *J. Inst. Maths. Applics.* 13, pp. 117–119.

Dantzig, G. B. (1963). *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey.

Dembo, R. S. and Steihaug T. (1980). Truncated-Newton algorithms for large-scale unconstrained optimization, Working paper #48, School of Organization and Management, Yale University.

Dennis, J. E., Jr. (1971). "Toward a unified convergence theory for Newton-like methods", in *Nonlinear Functional Analysis and Applications* (L. B. Rall, ed.), pp. 425–472, Academic Press, New York.

Dennis, J. E., Jr. and Schnabel, R. B. (1978). Least-change secant updates for quasi-Newton methods, Technical Report 78-344, Department of Computer Science, Cornell University.

Escudero, L. (1980). A projected Lagrangian method for nonlinear programming, Report G320-3401, IBM Palo Alto Scientific Center.

Faddeeva, V. N. (1959). *Computational Methods of Linear Algebra*, Dover Publications, New York.

Fiacco, A. V. and McCormick, G. P. (1968). *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York.

Fletcher, R. (1974). "Methods related to Lagrangian functions", in *Numerical Methods for Constrained Optimization* (P. E. Gill and W. Murray, eds.), pp. 219–240, Academic Press, London and New York.

Forrest, J. J. H. and Tomlin, J. A. (1972). Updating triangular factors of the basis to maintain sparsity in the product form simplex method, *Math. Prog.* 2, pp. 263–278.

Garcia-Palomares, U. M. and Mangasarian, O. L. (1976). Superlinearly convergent quasi-Newton algorithms for nonlinearly constrained optimization problems, *Math. Prog.* 11, pp. 1–13.

Gill, P. E. and Murray, W. (1974). Newton-type methods for unconstrained and linearly constrained optimization, *Math. Prog.* 28, pp. 311–350.

Gill, P. E. and Murray, W. (1979a). The computation of Lagrange multiplier estimates for constrained minimization, *Math. Prog.* 17, pp 32–60.

Gill, P. E. and Murray, W. (1979b). Conjugate-gradient methods for large-scale nonlinear optimization, Report SOL 79-15, Department of Operations Research, Stanford University.

Gill, P. E., Murray, W. and Nash, S. G. (1981). Newton-type minimization methods using the linear conjugate-gradient method, Report (to appear), Department of Operations Research, Stanford University, California.

Han, S.-P. (1976). Superlinearly convergent variable metric algorithms for general nonlinear programming problems, *Math. Prog.* 11, pp. 263–282.

Han, S.-P. (1977). A globally convergent method for nonlinear programming, *J. Opt. Th. Applics.* 22, pp. 297–310.

Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards* 49, pp. 409–436.

Hillstrom, K. E. (1977). A simulation test approach to the evaluation of nonlinear optimisation algorithms, *ACM Trans. Math. Software* 3, pp. 305–315.

Lemke, C. E. (1965). Bimatrix equilibrium points and mathematical programming, *Management Science* 11, pp. 681–689.

Lyness, J. N. and Greenwell, C. (1977). A pilot scheme for minimisation software evaluation, Tech. Memo. 323, Argonne National Laboratory, Argonne, Illinois.

Maratos, N. (1978). *Exact Penalty Function Algorithms for Finite-Dimensional and Control Optimisation Problems*, Ph. D. Thesis, University of London.

Murray, W. (1969). "An algorithm for constrained minimisation", in *Optimisation* (R. Fletcher, ed.), pp. 247–258, Academic Press, London and New York.

Murray, W. and Wright, M. H. (1980). Computation of the search direction in constrained optimisation algorithms, Report SOL 80-2, Department of Operations Research, Stanford University, to appear in *Math. Prog. Study on Constrained Optimisation*.

Murtagh, B. A. and Saunders, M. A. (1977). MINOS User's Guide, Report SOL 77-9, Department of Operations Research, Stanford University.

Murtagh, B. A. and Saunders, M. A. (1978). Large-scale linearly constrained optimisation, *Math. Prog.* 14, pp. 41–72.

Murtagh, B. A. and Saunders, M. A. (1980a). The implementation of a Lagrangian-based algorithm for sparse nonlinear constraints, Report SOL 80-1, Department of Operations Research, Stanford University, to appear in *Math. Prog. Study on Constrained Optimisation*.

Murtagh, B. A. and Saunders, M. A. (1980b). MINOS/AUGMENTED User's Manual, Report SOL 80-14, Department of Operations Research, Stanford University.

Orchard-Hays, W. (1968). *Advanced Linear-Programming Computing Techniques*, McGraw-Hill, New York.

Ortega, J. M. and Rheinboldt, W. C. (1970). *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, London and New York.

Powell, M. J. D. (1977). A fast algorithm for nonlinearly constrained optimization calculations, Report DAMTP 77/NA 2, University of Cambridge.

Powell, M. J. D. (1978). "The convergence of variable metric methods for nonlinearly constrained optimization calculations", in *Nonlinear Programming 3* (O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds.), pp. 27–63, Academic Press, London and New York.

Powell, M. J. D. and Toint, Ph. L. (1979). On the estimation of sparse Hessian matrices, *SIAM J. Numer. Anal.* 16, pp. 1060–1074.

Reid, J. K. (1976). Fortran subroutines for handling sparse linear programming bases, Report R8269, Atomic Energy Research Establishment, Harwell, England.

Robinson, S. M. (1972). A quadratically convergent algorithm for general nonlinear programming problems, *Math. Prog.* 3, pp. 145–156.

Robinson, S. M. (1974). Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming problems, *Math. Prog.* 7, pp. 1–16.

Rosen, J. B. (1978). "Two-phase algorithm for nonlinear constraint problems", in *Nonlinear Programming 3*, (O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds.), pp. 97–124, Academic Press, London and New York.

Rosen, J. B. and Kreuser, J. (1972). "A gradient projection algorithm for nonlinear constraints", in *Numerical Methods for Non-linear Optimization*, (F. A. Lootsma, ed.), pp. 297–300, Academic Press, London and New York.

Saunders, M. A. (1976). A fast, stable implementation of the simplex method using Bartels-Golub updating, in *Sparse Matrix Computations*, (J. R. Bunch and D. J. Rose, eds.), pp. 213–226, Academic Press, London and New York.

Shanno, D. F. (1978). Conjugate gradient methods with inexact searches, *Math. of Oper. Res.* 3, pp. 244–256.

Shanno, D. F. (1979). Computational experience with methods for estimating sparse Hessians for nonlinear optimization, Report MIS 79-8, School of Management and Information Science, University of Arizona.

Tapia, R. A. (1978). "Quasi-Newton methods for equality constrained optimisation: equivalence of existing methods and a new implementation", in *Nonlinear Programming 3*, (O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds.), pp. 125–164, Academic Press, London and New York.

Thapa, M. N. (1980). *Optimisation of Unconstrained Functions with Sparse Hessian Matrices*, Ph. D. Thesis, Stanford University.

Toint, P. L. (1977). On sparse and symmetric matrix updating subject to a linear equation, *Math. Comp.* 31, pp. 954–961.

Tomlin, J. A. (1976). Robust implementation of Lemke's method for the linear complementarity problem, Report SOL 76-24, Department of Operations Research, Stanford University.

Van der Hoek, G. (1979). Asymptotic properties of reduction methods applying linearly equality constrained reduced problems, Report 7933, Econometric Institute, Erasmus University, Rotterdam.

Wilkinson, J. H. (1965). *The Algebraic Eigenvalue Problem*, The Clarendon Press, Oxford.

Wilson, R. B. (1963). *A Simplicial Algorithm for Concave Programming*, Ph. D. Thesis, Harvard University.

Wolfe, P. (1962). The reduced gradient method, unpublished manuscript, The RAND Corporation.

Wright, M. H. (1976). *Numerical Methods for Nonlinearly Constrained Optimisation*, Ph. D. Thesis, Stanford University.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 81-1 ✓	2. GOVT ACCESSION NO. AD-A097 407	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) QP-Based Methods for Large-Scale Nonlinearly Constrained Optimization		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright		8. CONTRACT OR GRANT NUMBER(s) DAAG29-79-C-0110 ✓ N00014-75-C-0267
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE January 1981 13. NUMBER OF PAGES 23
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

large-scale optimization
QP-based methods
nonlinearly constrained optimization
Lagrangian methods
sparse Hessian
sparse linear and nonlinear constraints
numerical software

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Several methods for nonlinearly constrained optimization have been suggested in recent years that are based on solving a quadratic programming (QP) subproblem to determine the direction of search. Even for dense problems, there is no consensus at present concerning the "best" formulation of the QP subproblem. When solving large problems, many of the options possible for small problems become unreasonably expensive in terms of storage and/or arithmetic operations. This paper discusses the inherent difficulties of developing QP-based methods for large-scale nonlinearly constrained optimization, and suggests some possible approaches.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DATE
ILMED
-8